

S M M P

U S E R M A N U A L (2005 Release)

F. Eisenmenger, U.H.E. Hansmann, Sh. Hayryan, C.-K. Hu

C O N T E N T S

1. INTRODUCTION
2. INSTALLATION
3. LIMITATIONS
4. LIBRARIES AND PARAMETERS
5. INPUT FILES
 - (a) PDB File
 - (b) Sequence File
 - (c) Configuration File (Dihedrals)
6. FREQUENTLY USED FUNCTIONS AND SUBROUTINES
7. HOW TO RUN SMMP
8. HOW TO CITE SMMP
9. EXAMPLES
 - (a) Minimization of vacuum energy from a given configuration
 - (b) Structure determination by Simulated Annealing
 - (c) Calculation of multicanonical parameters
 - (d) Regularization of the B domain of protein A
 - (e) Parallel Tempering simulation of Met-enkephalin

1 Introduction

The SMMP (**S**imple **M**olecular **M**echanics for **P**roteins) package is designed for molecular simulations of linear proteins within the standard geometry model. The package contains various modern Monte Carlo algorithms and energy minimization routines. The energy of a protein can be calculated by exploiting one of three force fields: ECEPP/2, ECEPP/3 or FLEX. Two subroutines for approximating protein-solvent interactions by means of calculating the solvent accessible surface area of atomic groups are included. Nine sets of solvation parameters are available. All calculations are done with double precision, but the user can easily modify this option by changing the corresponding IMPLICIT statement.

This manual provides the user with the necessary information for installation of the program, to format input and output files, a brief description of the important routines, and describes some limitations of the package. A few example runs providing detailed explanation of input and output are added. More information on SMMP, definitions, background and references for force fields and algorithms as used in the program package can be found in [1,2].

SMMP is offered as open source code. We encourage users to re-write code or add their own modules. This task is eased up by the simplicity of the programs written in standard FORTRAN. As a courtesy to the authors and future users of SMMP we ask you to send us a note if you feel that your add-on or your modification of SMMP would be useful for other users, too. The usage and distribution of the program are governed by the following license agreement:

By viewing or using any part of the program package SMMP you accept the following terms:

SMMP is free software; it can be used modified and redistributed under the terms of the GNU General Public License (<http://www.gnu.org/licenses/gpl.html>) as published by the Free Software Foundation, EXCEPT WHERE RESTRICTED BY ONE OF THE FOLLOWING CLAUSES.

The program may be only used in the context of scientific research and teaching at a university or university-like institution. Its use for commercial and/or military purposes, or in the context of commercial and/or military research is explicitly forbidden.

The use of SMMP has to be acknowledged in all resulting publications by quoting: *F. Eisenmenger, U.H.E. Hansmann, S. Hayryan and C.K. Hu, [SMMP] A Modern Package for Protein Simulations, Comp.Phys. Comm. 138 (2001) 192-212; An Enhanced Version of SMMP - Open-Source Software Package for Simulation of Proteins, submitted.*

No part of SMMP shall be incorporated into other programs without explicit

permission of the authors.

SMMP is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

The most recent version of SMMP can be found on the program package's web sites: **<http://www.smmp05.net>** or **<http://www.phy.mtu.edu/biophys/smmp.htm>**. Any suggestions for improvements of the code or reports on bugs are welcome. Please send your remarks to:

Frank Eisenmenger
Forschungsinstitut für Molekulare Pharmakologie, 13125 Berlin, Germany.
E-mail: eisenmenger@fmp-berlin.de

Ulrich H.E. Hansmann
Michigan Technological University, Houghton, MI 49931-1291, USA.
E-mail: hansmann@mtu.edu

Shura Hayryan
Institute of Physics, Academia Sinica, Nankang, Taipei 11529, Taiwan
E-mail: shura@phys.sinica.edu.tw

Chin-Kun Hu
Institute of Physics, Academia Sinica, Nankang, Taipei 11529, Taiwan
E-mail: huck@phys.sinica.edu.tw

Acknowledgments:

U.H. acknowledges support by a research grant (CHE-9981874) of the National Science Foundation (USA). S.H. and C.K.H. are supported by Academia Sinica (Taipei) and by the National Science Council of the Republic of China (Taiwan) under Contract No. NSC 89-2112-M-001-084.

2 Installation

The complete SMMP package is written in standard FORTRAN-77 language. We have been exploiting it using the standard Gnu (g77) and the Intel® Fortran compiler for Linux, but it should be possible to compile and link the code with any contemporary FORTRAN compiler. All subroutines are stored in files *.f, and an example Makefile is included in the package. There are no machine dependent routines included in SMMP. All COMMON blocks and the limiting parameters are gathered in INCL.H, which is attached to the necessary modules through an INCLUDE statement. After uncompressing and unpacking (via 'tar -xvf' command) the SMMP package into a separate directory, the following files can be found in that directory:

- README A README-file
- SMMP A sub-directory containing:
 - manual.pdf The manual in PDF-format describing the details of various subroutines and the installation of SMMP
 - license.txt Ascii-file with the license agreement
 - lib.sh2 Library file with ECEPP/2 parameters
 - lib.sh3 Library file with ECEPP/3 parameters
 - lib.flex Library file with FLEX parameters
 - charges File with charges (needed only for FLEX)
 - tes.dat File with tessellation points for approximating the solvent accessible surface area of atoms
- Makefile Produces the executable file (sequential jobs)
- Makefile_p Produces the executable file for Parallel Tempering jobs on multiple-processor machines
- INCL.H Defines global parameters and common-blocks
- INCP.H Defines parameters and common-blocks necessary for reading PDB-structures
- init_molecule.f Wrapping function that constructs the start configuration of a molecule
- init_energy.f Wrapping function that initializes the energy parameters
- redseq.f Reads sequence of amino acids (e.g. enkefa.seq)
- redvar.f Reads initial configuration of molecule (e.g. enkefa.var)
- getmol.f Assembles data from libraries
- bldmol.f Builds up the protein atom coordinates
- addend.f Adds end-groups
- setmvs.f Determines sets of moving atoms for given variables
- mklist.f Compiles lists of interaction partners
- setvar.f Resets variables and rebuilds the molecule
- dihedr.f Returns dihedral angles and valence angles

- difang.f	Calculates the difference and the sum of two dihedral angles
- nursvr.f	Calculates the residue index of a dihedral angle or atom
- redstr.f	String input routines
- pdbread.f	Reads amino acid sequence and atomic coordinates from a PDB-file, calculates dihedral angles from PDB-coordinates, and builds index field that relates PDB-atoms to SMMP-atoms
- energy.f	Wrapping function that returns the energy of a current protein configuration
- enyflx.f	Calculates internal energy of molecule with FLEX dataset
- enyshe.f	Calculates internal energy of molecule with ECEPP datasets
- enysol.f	Calculates solvation energy of molecule using solvent accessible area method (fast, but approximate calculation)
- esolan.f	Calculates solvation energy of molecule using solvent accessible area method (analytic, but slow calculation)
- enyreg.f	Calculates a constraint energy needed for regularizing PDB-structures
- gradient.f	Wrapping function that returns the energy gradient vs. dihedral angles for a protein configuration
- opeflx.f	Calculates internal energy and partial derivatives vs. dihedral angles for FLEX dataset
- opeshe.f	Calculates internal energy and partial derivatives vs. dihedral angles for ECEPP datasets
- opesol.f	Calculates analytical partial derivatives vs. dihedral angles of the solvation energy
- opereg.f	Calculates the partial derivatives vs. dihedral angles of the constraint energy term during regularization
- main.f	Main program
- regul.f	Regularization of PDB-structure into SMMP geometry
- anneal.f	For simulated annealing run
- canon.f	For canonical Monte Carlo run
- minim.f	For minimization of protein potential energy
- mulcan_par.f	Calculates multicanonical weights
- mulcan_sim.f	For multicanonical simulation run
- partem_s.f	For parallel tempering run on a single-processor machine
- partem_p.f	For parallel tempering run on a multiple-processor machine
- pmain.f	Replaces 'main.f' on multiple-processor machine in parallel tempering runs.
- metropolis.f	Performs Metropolis updates
- minqsn.f	Minimization by quasi-Newton method
- mincjpg.f	Minimization by Conjugate Gradient method

- outvar.f Output of the current conformation
- contacts.f Calculates van der Waals contacts
- cnteny.f Calculates atomic contact energy and prints bad contacts
- hbond.f Calculates number of hydrogen bonds in a configuration
- helix.f Measures the number of residues which are part of alpha-helix or beta-sheet
- outpdb.f Output of current configuration in PDB-format
- rgyr.f Measures the radius-of-gyration and end-to-end distance in a molecule
- zimmer.f Expresses given configuration in Zimmerman code
- rmsdfun.f Calculates root-mean-square deviation between current SMMP configuration and a reference structure.

- EXAMPLES A sub-directory containing:
 - enkefa.seq Example sequence file
 - enkefa.var Example configuration file
 - enkefa.ann Example configuration file for simulated annealing run
 - enkefa.ref Example contact matrix file
 - enkefa.temp Example temperature file for parallel tempering runs
 - smmp.cmd Example shell-script to run smmp
 - 1bdd.pdb Example PDB-file (for regularization)
 - smmp.reg Example shell-script to run smmp for regularization

For the installation of SMMP the user needs to apply the Makefile in the directory with the source code. One may edit this Makefile and customize the compiler command and compiler options to be used. Executing the command 'make' will complete the installation of SMMP.

3 Limitations

All parameters which limit the usage of SMMP are stored in the file INCL.H. The most important ones are listed below. The values of these parameters should be changed only in a consistent way!

mxml=1	max. number of molecules
mxrs=100	max. total number of residues
mxat=2000	max. total number of atoms
mxbd=3	max. number of bonds to following atoms
mxvr=mxrs*5	max. number of local variables
mxms=mxvr*3	max. total number of moving sets
mxvw=mxat*4	max. number of vdw domains
mx14=mxat*4	max. number of '1-4' partners
mxath=100,	max. number of atoms in help-arrays
mxvrh=mxath	max. number of variables in help
mxyat=18	max. number of energetic atom-types
mxhbdo=4	max. types of Hydrogens as donors in HB
mxhbac=6	max. types of atoms as acceptors in HB
mxyto=19	max. number of types of torsional potentials
nrsty=35	max. number of residue types
mxtysol=9	the number of solvation parameters sets

Note also the following restrictions in the current version of SMMP:

1. Only single molecules can be studied, therefore the parameter MXML must be always set equal to 1,
2. A single amino acid residue can not be simulated with the FLEX potential,
3. A protein must not start with a prolyl residue.

4 Libraries and Parameter Files

The residues which can be used with each parameter set are described in files *lib.sh2*, *lib.sh3*, and *lib.flex*, respectively. The file *charges* is needed for N- and C-terminal residues with FLEX parameters. The name of the directory, containing these 4 files should be given in string 'libdir', which is assigned in module MAIN. The choice between parameter sets for potentials ECEPP/2 and ECEPP/3 is done by the logical variable 'sh2'. The potential FLEX is set by the logical variable 'flex'.

The amino acid residue libraries contain chemical and structural data for each residue according the IUPAC-IUB nomenclature. The library files consist of blocks of records with each block representing one residue. The first line in each block starts with a '#' and contains the name of the residue and its total number of atoms. Each following line within the block describes one atom. Listed are: its name, the bond length and the valence angle to construct this atom, the type and the name of corresponding torsion angle, the partial charge for the atom, the type of the atom, as well as the previous and the following atoms (max. 3), it is connected to. The atom and torsional parameters in the libraries are the same as in the original ECEPP/2/3 and FLEX potentials. However, it should be noted that SMMP has its own numeration of the atom types and the types of torsions (see [1,2] for more details).

The arrays with the parameters for non bonded pairwise potentials (Van der Waals, hydrogen bonds, electrostatic) and the atomic solvation parameters are stored in the BLOCK DATA section of 'init_energy.f'.

SMMP employs 9 different sets of atomic solvation parameters. We do not include references to the original works where these sets are defined in this document, since this manual is only a supplement to our articles [1,2] where all relevant references can be found. The set of solvation parameters chosen is indicated by the value of integer variable 'itysol' in the MAIN module. If setting *itysol* to a positive value between 1 and 9 in MAIN, the numerical method *enysol* is applied to compute the solvation energy term. By choosing a negative integer value for *itysol* from the interval [-1,-9], the analytical method *esolan* is used to calculate the solvation energy. The tessellation points for numerically calculating the solvent accessible surface area have to be provided in an external file (named *tes.dat* in our package) and read in during initialization of solvation parameters in *init_energy.f*.

5 Input Files

SMMP requires an input file that specifies the sequence of amino acid residues of the protein in plain ASCII-format. Note, that because of limitations in the present version of SMMP, only one molecule may be specified.

5.1 PDB-File

Setting the variable '*iabin* = 0' in *main.f* the sequence of amino acids is read from a PDB-file - the standard format in which protein structures are deposited in the Protein Data Bank. The atomic coordinates are also read from the PDB-file and can serve as a start configuration. However, one should remember that the fixed bond lengths and angles with ECEPP or FLEX potential will slightly differ from the actual bond lengths and angles in the PDB-structure. Forcing the molecule into standard bonding geometry corresponding to a given potential may lead to un-physically high energies. Regularization through *regul.f* allows to obtain an optimized structure with standard bonding geometry with low internal energy without differing significantly from the PDB structure.

5.2 Sequence File

Setting '*iabin* = 1' the amino acid sequence is read from a separate sequence file. Its first line must start with a '#' and may contain the name for the molecule. The residues in the following lines should be named as in the library files *lib.sh2*, *lib.sh3*, or *lib.flex*. Residue names are not case-sensitive and should be separated from each other by at least one white space.

Currently, the following residue types can be used in SMMP:

1. Neutral:
ala, arg, asn, asp, cys, gln, glu, gly, his* ; hise* ; hyp* , ile, leu, lys, met, phe, pro* , ser, thr, trp, tyr, val
(where the residues marked by an asterix are characterized by:
his - hydrogen at N-delta atom
hise - hydrogen at N-epsilon atom
pro, hyp - down-puckering)
2. Charged:
arg+, asp-, glu-, his+, lys+.
3. Variants, only with ECEPP/3 parameters:
hypu, prou - up-puckering
cpro, cpru - cis-Pro with down-, up-puckering
pron, pro+ - N-terminal neutral, charged with respect to NH2+ Pro

End-groups need to be specified in *main.f* and are added through subroutine *addend* to the N- and C-terminus, correspondingly. At present, molecules with only one single

residue cannot be simulated with the FLEX data set and the first residue should not be a prolyl residue.

An example sequence file (provided with SMMP) may look as follows:

```
----- enkefa.seq -----
# Met-Enkephalin
Tyr Gly Gly Phe Met
-----
```

5.3 Configuration File (Dihedrals)

With option '*iabin* = 1' SMMP allows to provide a designated start configuration by listing dihedral angles in a second input file. If no name of any configuration file is given (or the name of a non-existing file is entered), all variables retain their default values as defined in the libraries.

This configuration file has to follow a fixed structure in which each line can be considered as a COMMAND for subroutine '*redvar*'. The following syntax has to be used in each COMMAND: RESIDUE : VARIABLE : VALUE Hence, each line can consist of up to 3 FIELDS, separated by an ':'. In the first field the RESIDUE is selected through an INTEGER number which marks its position in the amino acid sequence. The second field should contain a string with the name of the VARIABLE, i.e. names the specific dihedral angle. The last field provides the value for the VARIABLE (a REAL number) and is mandatory. A symbol '&' following this number indicates that the variable will be fixed to that value throughout the whole simulation process. Spaces are not significant and are therefore ignored. Empty COMMANDS, as ' : : ', and empty lines or lines containing '#' are ignored. Missing fields are interpreted as 'for all':

```
1 : phi : 180          Set phi of residue #1 to 180 degrees
1 : phi : 180&         Keep phi of residue #1 fixed to 180 degrees
phi : 180              Set phi of ALL residues to 180 degrees
180                    Set all variables to 180 degrees
```

Several consecutive residues or variables can be indicated by ZONES of indices. Several NAMES can be indicated by wild-card ('*') and are case-sensitive:

```
1-4 : phi : 180        Set 'phi' of residues #1,#2,#3, and #4 to 180 deg.
5 : x* : 60             Set all xi-angles of residue #5 to 60 deg.
```

Several NAMES or INDICES may be given in the same field, when separated by ','. Similarly, several commands may be given on the same line and must be separated by ';'.

```
phi, psi : -30         Set all phi & psi to -30 deg.
phi:-65; psi:-45       Set all phi=-65, all psi=-45
```

Note, that dihedral angles are defined slightly differently in SMMP than in standard ECEPP (residue index in parenthesis):

- psi of the i-th residue = defined by $N(i-1) - CA(i-1) - C(i-1) - N(i)$
- omg of the i-th residue = defined by $CA(i-1) - C(i-1) - N(i) - CA(i)$
- phi for the i-th residue = defined by $C(i-1) - N(i) - CA(i) - C(i)$

except:

phi for the 1st residue = defined by $H1(1) - N(1) - CA(1) - C(1)$

(H2-atom in the N-terminal NH2-group is added via dihedral 'H2-N-CA-C'= phi + 120)

If a molecule consists of n residues then:

- pst = defines dihedral $N(n) - CA(n) - C(n) - Oxt(n)$
- omt = defines dihedral $CA(n) - C(n) - Oxt(n) - Hxt(n)$

(Oxt & Hxt comprise the hydroxyl of the C-terminal carboxyl group)

The following example configuration file *enkefa.var* is provided with the program and may be used with the sequence file *enkefa.seq*:

```
----- enkefa.var -----
1 : phi : -86.24
1 : x1 : -172.59
1 : x2 : 78.71
1 : x6 : -165.88
2 : psi : 156.18
2 : omg : 180.00
2 : phi : -154.53
3 : psi : 83.64
3 : omg : 180.00
3 : phi : 83.66
4 : psi : -73.86
4 : omg : 180.00
4 : phi : -137.04
4 : x1 : 58.79
4 : x2 : 94.60
5 : psi : 19.33
5 : omg : -180.00
5 : phi : -163.63
5 : x1 : 52.76
5 : x2 : 175.28
5 : x3 : -179.83
5 : x4 : -58.57
5 : pst : 160.45
5 : omt : 180.00
-----
```

6 Frequently used Functions and Subroutines

6.1 Task Subroutines

In the following we describe some important simulation subroutines and their requisites. For more information see [1,2]. Following the philosophy behind SMMP these routines are rather simple and may serve as templates which the user can modify according to his special needs.

- Subroutine REGUL (file *regul.f*)
(Usage: *'call regul(nml)'*)

For a given PDB-structure the actual bond lengths and bond angles generally differ from those in the standard geometry model assumed with the ECEPP or FLEX potential. This subroutine is applied to determine the configuration within the standard geometry that is as close as possible to a given PDB-structure with an energy as low as possible. This process of fitting a PDB structure into standard geometry is called regularization. *'regul'* requires the PDB-structure to be read into SMMP by setting *'iabin = 0'* in *'main'*. With this option, *'init_molecule'* calls the subroutines *'pdbread'* and *'pdbvars'* that also to measure the dihedral angles in the PDB-structure. Using these dihedral angles a first model of the molecule is built using standard bonding geometry. Hence, *'regul'* has to be called AFTER *'init_molecule'*. *'nml'* has to be set to 1 in the present version of SMMP.

'regul' starts with a minimization of the initial structure using only the sum of squared distances between atom positions of the SMMP structure to the PDB-structure as "constraint energy". After this initial step, a list of bad atom contacts, i.e. atoms with vdW-energy of more than 2 kcal/mol (as calculated by *'cnteny'*) and the root-mean-square-deviation (rmsd) to the PDB-structure is printed out. In a second step, the physical energy is minimized, allowing only hydrogens to move, and bad contacts and the rmsd are evaluated, again. A number of iterations follow where a composite energy as the weighted sum of physical and constraint energy is minimized. In this iterations the weight of the constraint energy is successively lowered to zero, and that of the physical energy raised to one. At the end, the dihedral angles of the final structure are printed out together with its rmsd and list of remaining bad contacts.

- Subroutine MINIM (file *minim.f*):
(Usage: *'call minim(imin)'*)

This subroutine minimizes the current configuration of the molecule. It can be called from any point during the simulation once a configuration is defined. When calling this subroutine one needs to specify the type of minimizer through variable *'imin'*. Setting *'imin = 1'* chooses a Quasi-Newton minimizer by calling *'minqsn'*, *'imin = 2'* a Conjugate Gradient minimizer (*'mincvg'*). The performance of these minimizers can be controlled by setting various parameters in MINIM, such as the accuracy (*'acc'*) or the total number of function calls *'mxop'* (see the header of MINIM for the various parameters that can be adjusted).

- Subroutine CANON (file *canon.f*):
The subroutine carries out a canonical simulation at temperature 'temp' which has to be assigned in a PARAMETER statement. The simulation starts from a random configuration if the logical parameter 'lrnd' is set to *.TRUE.*. Otherwise the simulation starts from the current conformation of dihedral angles as stored in array 'vlvr'. In separated PARAMETER statements one also has to set the number of Monte Carlo sweeps for the simulation, 'nswp', and at the equilibration, 'nequi', as well as the number 'nmes' of Monte Carlo sweeps between measurements. Three output files are created: files 'start.pdb' and 'final.pdb' are for storing the start and final configurations of the molecule, respectively, in a data format compatible with Protein Data Bank. The file 'time.d' contains the time series of some quantities. In our template we measure and store in this file the total energy 'eol'; the radius-of-gyration 'rgy' (by means of subroutine RGYR); the number of helical (beta sheet-like) residues 'nhel' ('nbet') and segments 'mhel' ('mbet') using subroutine HELIX; and the number of hydrogen bonds 'mhb' calculated through subroutine HBOND.
- Subroutine ANNEAL (file: *anneal.f*):
This subroutine allows for optimization of protein conformations by means of simulated annealing. When the logical parameter 'lrnd' is set to *.TRUE.* then the program will start from a random configuration. Otherwise, the annealing process will start from the current configuration as stored in the array 'vlvr'. In PARAMETER statements the user has to set the initial and final temperatures 'tmax' and 'tmin', the number of Monte Carlo sweeps for simulation NSWP and equilibration 'nequi', and the number 'nmes' of Monte Carlo sweeps between measurements. Four output files are created in the same directory where the executable is running: three of them contain the start (start.pdb), the final (final.pdb) and the global minimum configuration (global.pdb), respectively, the 4th file (time.d) contains the time series of the simulation. In the presented template we measure and store in this file the actual temperature 'temp', the total energy 'eol', the radius-of-gyration 'rgy' (as calculated by subroutine RGYR) and the Zimmerman code (as character variable 'zimm') of the present configuration as obtained through subroutine ZIMMER.
- Subroutine MULCAN PAR (file *mulcan-par.f*):
This subroutine calculates multicanonical weight factors and has two working modes. One is for improving the already existing set of multicanonical parameters through further iterations, and the another is to calculate them anew. In the first case the logical variable 'litr' is set to the value *.TRUE.* and the input information is the histogram from a preliminary simulation as well as the existing values of the parameters are read from the file 'mpar_full.d'. In the second case ('litr' = *.FALSE.*) all these arrays are initialized to zero. In PARAMETER statements lower and upper limit 'kmin' and 'kmax' of the energy range and the size of the energy bin 'ebin' have to be set. Other parameters which one has to set are: the total number of Monte Carlo sweeps 'nsweep', the number of Monte Carlo sweeps 'nup' in between updates of multicanonical parameters, and the temperature 'temp' of the initial canonical simulation. Every 'nup' sweeps the histogram of the preliminary

simulation as well as the existing values of the parameters are written to the file 'mpar_full.d'. The final multicanonical parameters are written into the file 'muca.d'.

- Subroutine MULCAN_SIM (file *mulcan_sim.f*):

This subroutine performs a Monte Carlo simulation a protein using multicanonical weights. Information necessary for a re-start (chosen by setting 'restart'=.TRUE.) - otherwise the simulation starts from a random configuration - are saved after every 'nsave' Monte Carlo moves into the file 'start.d'. Parameters 'ebin', 'kmin', and 'kmax' set the energy bin size and the lower and upper limits of the energy interval, within which the multicanonical parameters were defined (see MULCAN_PAR). These parameters are read from the file 'muca.d' and have to be the same as the ones used in MULCAN_PAR to generate the multicanonical weights. Other parameters which one has to initialize are: the total number of Monte Carlo sweeps 'nsweep', the number of Monte Carlo sweeps 'nequi' needed for equilibration, and the number of Monte Carlo sweeps 'nmes' between successive measurements. In our example the output goes to file 'time.d', in which the time series of the simulation is written. Measured quantities are the total potential energy 'eol', the number of contacts 'nhx' and the number of native contacts 'nhy'. The latter number counts for the contacts which are same in the present configuration and the target structure. Also measured is the Hamming distance 'dham' between the present and the target configuration. The latter three quantities are calculated by the subroutine CONTACTS. A call of this module requires the preparation of a file 'reference.d' with the contact matrix of the target structure.

- Subroutine PARTEM_S (file *partem_s.f*):

This parameterless subroutine is designed to simulate a protein by means of the parallel tempering technique on a single processor. Before running a parallel tempering simulation using PARTEM_S one has firstly to set the following variables in this subroutine: the number of replicas ('no') and the initial temperatures on each replica (array 'temp(no)'); the sum of the number of degrees of freedom over all replicas ('nvrmax'); the number of Monte Carlo sweeps at equilibration ('nequi') and the number of Monte Carlo sweeps in-between measurements ('nmes'). If not for re-starts, the logical variable 'lstart' should be set to .FALSE., in which case all dihedral angles are set to random values. At the end of the simulation all dihedral angles are stored in the file 'startconf.d', from which the program reads its start configuration, if 'lstart=.TRUE.'. In the simulation, measurements are made every 'nmes' Monte Carlo sweeps. In our template, we measure for each replica the radius-of gyration 'rgy' and the end-to-end distance 'ee' by means of subroutine RGYR and write both values together with the temperature 'temp0', total energy 'energy', Monte Carlo time 'nsw', and the index of the replicas 'k1' to an output file 'time.d'.

- PMAIN and subroutine PARTEM_P (files *pmain.f* and *partem_p.f*):

These subroutines are used for parallel tempering simulations on a parallel computer with 'no' nodes. PMAIN replaces MAIN in this case. Both PMAIN and PARTEM_P

utilize the MPI message-passing routines. The number of nodes 'no' has to be set in both modules and equals to the number of replicas in the parallel tempering method. In addition, one has to set in PARTEM_P also the following variables: the number of Monte Carlo sweeps for equilibration ('nequi') and simulation ('nsweep'); and the number of Monte Carlo sweeps between measurements ('nmes'). If not for re-starts the logical variable 'newsta' should be set to .FALSE., in which case all dihedral angles are set to the random values. At the end of the simulation all dihedral angles are stored in the file 'par R.in', from which the program reads its start configurations if 'newsta=.TRUE.'. Before running the program one has to prepare the file 'temp.d' with 'no' lines. Each line lists the index of replica 'j' and its temperature 'temp'. In our template we measure every 'nmes' Monte Carlo sweeps for each replica a number of quantities: the radius of gyration 'rgy', the number of residues 'nhel' and segments 'mhel' in an α -helix, the number of residues in a β -sheet 'nbet' and the number of such segments 'mbet', the number of hydrogen bonds 'mhb', and the number of contacts 'nctot' and native contacts 'ncnat'. The later number counts the contacts which are same in the present configuration and in the target structure. For this, one has to prepare in advance the file 'reference.d' with the contact matrix of the target structure. All measured quantities are written together with energy, temperature and the index of the corresponding replica in the file 'ts.d'. At the end of the simulation the average energy and specific heat for each temperature/replica and the acceptance rate of replica-exchange moves are written to standard output.

6.2 Useful Subroutines

- Subroutine CONTACTS

(Usage: *call contacts(ncn,nham2,dham)*):

This subroutine calculates the matrix of the van-der-Waals contacts between C $^{\alpha}$ -atoms for a given configuration. 'ncn' is the total number of such contacts, 'nham2' the number of native contacts (i.e. such contacts which are the same in the present configuration and the reference configuration), and the Hamming distance 'dham' between the present and the target configuration.

- Subroutine CNTENY:

(Usage: *call cnteny(nml)*)

The subroutine 'cnteny' displays a list of heavy atoms in molecule 'nml' that have a contact energy of more than 2 kcal/mol. If one sets the parameter 'ieyel = 0' in 'cnteny' the the contact energy is calculated with the van-der-Waals energy term, only. Setting 'ieyel = 1' additionally includes the electrostatic energy into the contact energy, and bad contacts are displayed if this energy exceeds a value of 10 kcal/mole for an atom.

- Functions ENERGY, ENYSHE, ENYFLX, ENYSOL, ESOLAN, and ENYREG:

(Usage: *eny = energy()*)

The real*8 function 'energy' is a wrapping function that calculates the energy of a

given configuration. Depending on the choice of parameters '*flex*', '*sh2*', '*itysol*' and '*ireg*' in '*main*' ('*pmain*') the 'vacuum' energy of the protein is calculated by the functions '*enyshe*' (ECEPP/2 or ECEPP/3), or '*enyflex*' (FLEX), and the solvation energy using the solvent accessible area method with '*enysol*' (approximate, but fast estimation of the solvated area) or '*esolan*' (analytical, but slower calculation of the area). '*enyreg*' calculates the constraint energy term needed during regularization of PDB-structures. '*energy*' returns the sum of these energy terms as selected by the parameters in the '*main*'-module.

- Subroutine GRADIENT, OPESHE, OPEFLX, OPESOL, and OPEREG:

(Usage: *call gradient()*)

The wrapping subroutine '*gradient*' calculates the energy and partial derivatives (with respect to internal degrees of freedom) for a given protein configuration. Depending on the choice of parameters '*flex*', '*sh2*', '*itysol*', and '*ireg*', the subroutines '*opeshe*' (ECEPP/2 or ECEPP/3), '*opeflx*' (FLEX), '*opesol*' (solvent accessible surface area) and/or '*opereg*' (constraint energy term during regularization of PDB structures) are selected and contribute to the total energy and its gradient that is calculated.

- Subroutine HBOND

(Usage: *call hbond(nml,mhb,ipr)*):

This subroutine determines the number '*mhb*' of the hydrogen bonds for the current conformation of the molecule and requires '*nml*' as input - the index of the present molecule (in current version of SMMP this variable has to be set to 1). The hydrogen bonds are printed out for '*ipr* > 0'.

- subroutine HELIX

(Usage: *helix(nhel,mhel,nbet,mbet)*):

HELIX determines whether a residue is part of an α -helix or β -sheet according to its backbone dihedral angles (ϕ, ψ) (ranges defined in a PARAMETER statement) and returns the number of helical residues '*nhel*', the number of helical segments '*mhel*', and the corresponding quantities for β -strand elements '*nbet*' and '*mbet*'.

- Subroutine METROPOLIS

(Usage: *call metropolis(eol,acz,weight)*):

The subroutine implements one Metropolis move for every dihedral angle and requires as input a weight function '*weight(eol)*' and the energy '*eol*' of the present configuration. It returns in the variable '*eol*' the energy of the configuration after '*nvr*' (the number of non-fixed dihedral angles) updates. '*acz*' accounts for the number of Metropolis moves were the proposed configuration was accepted.

- Subroutine OUTPDB

(Usage: *call outpdb(nml,npdb)*):

The subroutine writes in the output file the coordinates of the atoms of given molecule. The format is compatible with the standard data representation format of the Brookhaven Protein Data bank. This subroutine requires as input the

index 'nml' of the given molecule (has to be equal to 1 in the present version of SMMP) and the logical unit-number 'npdb' of the output-file.

- Subroutine OUTVAR

(Usage: *call outvar(nml,iswitch)*):

This subroutine writes the dihedral angles of the current configuration of molecule 'nml' to either standard output ('iswitch ≤ 0') or a dedicated file ('iswitch > 0', where *iswitch* is the logical unit number of the file). The dihedral angles are written out in a form that allows to use them as configuration input file for SMMP.

- Function RAND:

Our package does not include a random generator. Hence, the user has to provide a random generator for calculating random numbers uniformly distributed in the interval [0,1] and has to modify accordingly the subroutines: METROPOLIS, CANON, ANNEAL, MULCAN_PAR, MULCAN_NEW, PARTEM_S, PARTEM_P, and ESOLAN. In the provided versions of these subroutines we have assumed that the default FORTRAN random-generator is used which, however, may cause some systems problems.

- Subroutine RGYR

(Usage: *call rgyr(nml,rgy,ee)*):

This subroutine calculates the radius-of-gyration 'rgy' and the end-to-end distance 'ee' for the protein molecule 'nml'. Both quantities characterize the compactness of a protein conformation.

- Function RMSDFUN

(Usage: *rmsd = rmsdfun(nml,ir1,ir2,ixat,xrf,yrf,zrf,isl)*)

The real*8 function '*rmsdfun*' calculates the root-mean square-deviation (rmsd) between atom positions in the current configuration of molecule 'nml' for a range of residues '*[ir1,ir2]*' and a reference structure given by arrays of atom coordinates '*xrf,yrf,zrf*'. For each atom position in the current SMMP-structure the array '*ixat*' provides the index of the equivalent atom in the reference structure, or yields a value of '0' if there is no equivalent. By setting '*isl=0*' the rmsd is calculated over all non-hydrogen (heavy) atoms, for '*isl = 1*' for backbone atoms only, and for '*isl = 2*' the rmsd is calculated only between C^α-atoms. The routine '*rmsdfun*' may only be called **after** an initial call of the subroutine '*rmsinit(nml,string)*'. If *string* = '*smmp*' the reference configuration is a SMMP configuration, otherwise it is read in from a PDB-file which name given by '*string*'.

- Subroutine ZIMMER

(Usage: *call zimmer(nresi)*):

This subroutine analyzes the backbone dihedrals of a given conformation and characterizes the state of each pair of (ϕ, ψ) by its Zimmerman code. The subroutine requires as input the number of residues 'nresi' and returns the character string 'zimm' with the Zimmerman code of the present configuration (through a common block).

7 How to Run SMMP

SMMP does not include an interpreter of user-defined commands. The preparation of a simulation must be done in the MAIN module calling corresponding simulation subroutine(s). After changing of these modules SMMP has to be re-compiled. We recommend that new users spend some time studying the structure of the '*main*', '*init_energy*', '*init_molecule*', and '*INCL.H*' before start using SMMP. The following steps summarize how to run SMMP:

1. Assign to the character variable 'libdir' the path to the directory containing the standard amino acid residue libraries and the file 'charges'.
2. Select the force field and solvation model by setting the four parameters 'flex', 'sh2', 'epsd', and 'itysol' to their appropriate values:

```
* flex=.TRUE.   : Flex potential,      flex=.FALSE.: ECEPP potential
* sh2 =.TRUE.   : ECEPP/2 potential, sh2=.FALSE.: ECEPP/3
* epsd=.TRUE.   : Distant dependent dielectric permittivity
* epsd=.FALSE. : epsilon = 2
* itysol = 0    : gas phase,
* itysol > 0    : approximation of protein-solvent interactions by means
                  of a solvent accessible surface area approach with
                  numerical estimation of the accessible area,
* itysol < 0    : same as above, but the accessible area is calculated
                  analytically (considerably than itysol > 0).
```

3. call *init_energy(libdir)* to initialize the energy parametrization
4. Choose the N-terminal and C-terminal groups by assigning to appropriate strings to variables 'grpn' and 'grpc'.
5. Choose how the initial input is read in:

```
* iabin = 0    : read from PDB-file
* iabin = 1    : read from sequence (and configuration) file
```

6. Enter the names of the corresponding file(s) '*seqfil*' and '*varfil*'. In the current version of this requested through an interactive command line dialog, issued by '*call init_molecule*'. The user can easily just supply the corresponding file names in a "here document", as shown in the example scripts.
7. At this point the program is ready for calling task subroutines. In the provided version of '*main*' the energy minimization subroutine is called through '*call minim*'. Detailed examples for this and other simulation tasks can be found in the following

section. Normally, simulation subroutines write data in output files, but one can also use output routines such as '*outpdb*' in '*main*'. The minimal output (written into standard output) is the name of the sequence file (extension *.seq*), name of configuration file (extension *.var*), and for each residue a list of dihedral angles together with their initially assigned values.

8. For parallel tempering jobs on a multiprocessor system one has to replace '*main*' by '*pmain*'. The above protocol still applies, but one has to provide in addition the number of nodes with the variable '*no*'.

8 How to cite SMMP

Use of SMMP should be acknowledged by quoting the following reference:

1. F. Eisenmenger, U.H.E. Hansmann, Sh. Hayryan, C.-K. Hu, **[SMMP] A Modern Package for Simulation of Proteins**, *Comp. Phys. Comm.* **138** (2001) 192-212.
2. F. Eisenmenger, U.H.E. Hansmann, Sh. Hayryan, C.-K. Hu, **An Enhanced Version of SMMP - Open-Source Software Package for Simulation of Proteins**, *submitted*.
3. Where appropriate, the program package's web-site should be quoted as:
<http://www.smmp05.net>.

9 Examples

9.1 Minimization of vacuum energy from a given configuration

The example MAIN module, has the following settings:

```
flex = .false.  
sh2 = .false.  
epsd = .false.  
itysol = 0  
grpn = 'nh2'  
grpc = 'cooh'  
iabin = 1
```

which restricts energy calculations to evaluation of the ECEPP/3 energy (no protein-solvent interactions taken into account), choses *nh2* as N-terminal group and at the C-terminus the *cooh*-group, and finally states that the input is read from a sequence/configuration file. The the simulation task subroutine is called, in our example:

```
call minim(1)
```

i.e. the simulation task is the local minimization of the vacuum energy of a configuration using the Quasi-Newton algorithm. After setting the directory path in variable '*libdir*' and compilation, SMMP with this default settings will minimize the configuration given in *enkefa.var* using the following script (file *smmp.cmd* (in sub-directory EXAMPLES)

```
cd ..  
./smmp<<!  
./EXAMPLES/enkefa.seq  
./EXAMPLES/enkefa.var  
!
```

Running SMMP with this script will lead to the following output:

```
file with SEQUENCE: ./EXAMPLES/enkefa.seq
```

```
file with VARIABLES: ./EXAMPLES/enkefa.var
```

```
redvar> Met-Enkephalin: residue    1 Tyr : x1  set    -172.590  
redvar> Met-Enkephalin: residue    1 Tyr : x2  set      78.710  
redvar> Met-Enkephalin: residue    1 Tyr : x6  set   -165.880  
redvar> Met-Enkephalin: residue    1 Tyr : phi set    -86.240  
redvar> Met-Enkephalin: residue    2 Gly : psi set    156.180  
redvar> Met-Enkephalin: residue    2 Gly : omg set    180.000  
redvar> Met-Enkephalin: residue    2 Gly : phi set   -154.530  
redvar> Met-Enkephalin: residue    3 Gly : psi set     83.640  
redvar> Met-Enkephalin: residue    3 Gly : omg set    180.000
```

```

redvar> Met-Enkephalin: residue    3 Gly : phi set      83.660
redvar> Met-Enkephalin: residue    4 Phe : psi set     -73.860
redvar> Met-Enkephalin: residue    4 Phe : omg set     180.000
redvar> Met-Enkephalin: residue    4 Phe : x1 set      58.790
redvar> Met-Enkephalin: residue    4 Phe : x2 set      94.600
redvar> Met-Enkephalin: residue    4 Phe : phi set    -137.040
redvar> Met-Enkephalin: residue    5 Met : psi set      19.330
redvar> Met-Enkephalin: residue    5 Met : omg set     180.000
redvar> Met-Enkephalin: residue    5 Met : x1 set      52.760
redvar> Met-Enkephalin: residue    5 Met : x2 set     175.280
redvar> Met-Enkephalin: residue    5 Met : x3 set    -179.830
redvar> Met-Enkephalin: residue    5 Met : x4 set     -58.570
redvar> Met-Enkephalin: residue    5 Met : phi set    -163.630
redvar> Met-Enkephalin: residue    5 Met : pst set     160.450
redvar> Met-Enkephalin: residue    5 Met : omt set     180.000

```

Energy BEFORE minimization:

Total: 0.10354E+04

Coulomb: 0.1991E+02 Lennard-Jones: 0.1142E+03 HB: 0.9008E+03

Variables: 0.4511E+00 Solvation: 0.0000E+00

```

Step    1: energy 0.489976E+05 ( 0.867243E+13 )
Step    2: energy 0.108241E+03 ( 0.964364E+07 )
Step    3: energy -0.632755E+00 ( 0.498256E+04 )
Step    4: energy -0.739539E+01 ( 0.109979E+05 )
Step    5: energy 0.517957E+03 ( 0.678892E+09 )
Step    6: energy -0.767774E+01 ( 0.143880E+05 )
Step    7: energy -0.818570E+01 ( 0.100364E+05 )
Step    8: energy -0.938401E+01 ( 0.172508E+04 )
Step    9: energy -0.109799E+02 ( 0.105718E+04 )
Step   10: energy -0.116244E+02 ( 0.580821E+03 )

```

.....

.....

.....

```

Step   80: energy -0.124285E+02 ( 0.370905E-08 )
Step   81: energy -0.124285E+02 ( 0.324300E-08 )
Step   82: energy -0.124285E+02 ( 0.777363E-09 )
Step   83: energy -0.124285E+02 ( 0.443358E-10 )
Step   84: energy -0.124285E+02 ( 0.322695E-11 )
Step   85: energy -0.124285E+02 ( 0.144369E-11 )
Step   86: energy -0.124285E+02 ( 0.322695E-11 )

```

---- CONVERGENCE ----

Final energies -----

Total: -0.12429E+02

Coulomb: 0.2143E+02 Lennard-Jones: -0.2923E+02 HB: -0.6706E+01

Variables: 0.2084E+01 Solvation: 0.0000E+00

Variables -----

x1	1	-173.2	(0.6)
x2	1	79.3	(0.6)
x6	1	-166.3	(0.5)
phi	1	-83.1	(3.2)
psi	2	155.8	(0.4)
omg	2	-177.1	(2.9)
phi	2	-154.2	(0.3)
psi	3	85.8	(2.2)
omg	3	168.5	(11.5)
phi	3	83.0	(0.7)
psi	4	-75.0	(1.2)
omg	4	-170.0	(10.0)
x1	4	58.9	(0.1)
x2	4	94.5	(0.1)
phi	4	-136.8	(0.2)
psi	5	19.1	(0.2)
omg	5	-174.1	(5.9)
x1	5	52.9	(0.1)
x2	5	175.3	(0.0)
x3	5	-179.9	(0.0)
x4	5	-58.6	(0.0)
phi	5	-163.4	(0.2)
pst	5	160.8	(0.3)
omt	5	-179.8	(0.2)

Gradient -----

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

9.2 Structure Determination by Simulated Annealing

For the following simulated annealing run for Met-Enkephalin defined through its sequence file *enkefa.seq*, another configuration file *enkefa.ann* is used. By this configuration file:

```
om*: 180.0 &
```

the dihedral angles *omg* and *omt* angles are all set to 180.0 degrees and fixed, subsequently. The reason for this choice is that simulated annealing performs only poorly for the case where these angles are free to change. Simulated annealing is chosen as simulation task by replacing

```
call minim(1)
```

through

```
call anneal
```

in *main*. In this example we further choose as force field ECEPP/2 (setting *sh2* = *.true.* and no protein-solvation interaction. Hence, we have the following settings for parameters in *main.f*:

```
flex = .false.  
sh2 = .true.  
epsd = .false.  
itysol = 0  
grpn = 'nh2'  
grpc = 'cooh'  
iabin = 1
```

Our simulated annealing run will consist of 100 Monte Carlo sweeps at highest temperature 1000 K followed by 100000 sweeps in which the temperature is gradually lowered down to a final temperature of 100 K. The progress of the simulation is monitored by writing data into the file '*time.d*', after every 1000 sweeps. The simulation shall start from a random configuration. This leads to the following settings in the PARAMETER statements of *anneal*:

```
parameter(lrand=.true.)  
parameter(nequi=100, nswp=100000,nmes=1000)  
parameter(tmax=1000.0,tmin=100.0)
```

After compilation of SMMP with the 'make' command and running the program from the sub-directory *EXAMPLES* by the script:

```
cd ..  
./smmp<<!  
./EXAMPLES/enkefa.seq  
./EXAMPLES/enkefa.ann  
!
```

the output will contain the energy of the true start configuration, energy after equilibration, the average acceptance rate of moves in the simulated annealing run, the final configuration (and its energy) and the configuration with the lowest energy found in this run:

file with SEQUENCE: ./EXAMPLES/enkefa.seq

file with VARIABLES: ./EXAMPLES/enkefa.ann

```
redvar> Met-Enkephalin: residue    2 Gly : omg set    180.000    Fixed
redvar> Met-Enkephalin: residue    3 Gly : omg set    180.000    Fixed
redvar> Met-Enkephalin: residue    4 Phe : omg set    180.000    Fixed
redvar> Met-Enkephalin: residue    5 Met : omg set    180.000    Fixed
redvar> Met-Enkephalin: residue    5 Met : omt set    180.000    Fixed
redvar> Molecule Met-Enkephalin:   19 variable(s) remain unchanged
```

energy of start configuration: 0.34360E+08

Energy after equilibration: 24.03751
acceptance rate: 0.186481819

last energy -6.56133682

```
1 : x1 :    173.7
1 : x2 :   -106.4
1 : x6 :    161.6
1 : phi :   -88.5
2 : psi :    153.2
2 : omg :    180.0  &
2 : phi :  -168.5
3 : psi :     87.3
3 : omg :  -180.0  &
3 : phi :     78.1
4 : psi :   -70.1
4 : omg :    180.0  &
4 : x1 :    174.3
4 : x2 :   -127.1
4 : phi :  -107.0
5 : psi :   -48.7
5 : omg :    180.0  &
5 : x1 :     67.6
5 : x2 :   -176.2
5 : x3 :    177.9
5 : x4 :    168.9
5 : phi :  -159.4
```

```

5 : pst :    174.3
5 : omt :   -180.0  &

```

lowest energy ever found: 55794 -9.61510136

```

1 : x1  :    173.7
1 : x2  :   -112.3
1 : x6  :    141.4
1 : phi :    -74.2
2 : psi :    152.8
2 : omg :    180.0  &
2 : phi :   -161.7
3 : psi :     65.3
3 : omg :   -180.0  &
3 : phi :     70.2
4 : psi :    -88.0
4 : omg :    180.0  &
4 : x1  :    178.2
4 : x2  :     76.5
4 : phi :    -87.9
5 : psi :    -29.9
5 : omg :    180.0  &
5 : x1  :    -61.9
5 : x2  :    171.5
5 : x3  :   -178.5
5 : x4  :    179.7
5 : phi :    -79.8
5 : pst :    141.8
5 : omt :   -180.0  &

```

The random start configuration, final configuration and lowest-energy configuration are also written in PDB-format into the files *start.pdb*, *final.pdb* and *global.pdb* which allows an easy visualization of these configurations with programs graphically displaying PDB-structures. The progress of the simulated annealing run is monitored through the time series in file *time.d* which list the Monte Carlo sweep, temperature, ECEPP/2 energy, radius-of-gyration and the Zimmerman-coding of the present configuration:

0	1000.000	19.256	5.982 EfbE
1000	977.237	12.361	6.020 EeaA
2000	954.993	22.999	5.234 DFbA
3000	933.254	18.473	6.072 GAHD
4000	912.011	13.257	6.199 bDhF
5000	891.251	18.902	4.951 DBAF
6000	870.964	14.526	5.144 eDDB
7000	851.138	11.837	5.299 EhdE
8000	831.764	15.603	4.821 FDdC

9000	812.831	10.447	5.118 GAaF
10000	794.328	19.689	5.710 EddA
.....			
.....			
.....			
90000	125.893	-5.295	4.670 FDcF
91000	123.027	-1.203	4.758 FDcF
92000	120.226	-5.452	4.677 FDcF
93000	117.490	-5.705	4.640 EDcF
94000	114.815	-4.723	4.598 FDcF
95000	112.202	-5.227	4.681 EDcF
96000	109.648	-6.126	4.611 FDcF
97000	107.152	-5.658	4.633 EDcF
98000	104.713	-5.066	4.552 EDcG
99000	102.329	-5.839	4.609 FDcG
100000	100.000	-6.561	4.536 FDcA

9.3 Calculation of Multicanonical Parameters

In the following example, multicanonical parameters are calculated for Met-enkephalin (with fixed omega angles) in gas-phase simulations relying on the ECEPP/3 force-field. For this purpose one has to replace in 'main' line 50

```
call minim(1)
```

through

```
call mulcan_par
```

and set the following parameters in 'main' accordingly:

```
flex = .false.  
sh2 = .false.  
epsd = .false.  
itysol = 0  
grpn = 'nh2'  
grpc = 'cooh'  
iabin = 1
```

The simulation has a total of 100,000 Monte Carlo sweeps and the multicanonical parameters are updated every 2,000 sweeps. The parameters are calculated from scratch, and we attempt to obtain a histogram in energy in the interval [-12,20] Kcal/mol with energy bin size 1 Kcal/mol. We expect that at temperature 1000 K we are clearly in the high-energy phase where successive configurations are little correlated. This leads to the following setting of parameters in subroutine mulcan par:

```
parameter(l_iter=.false.)  
parameter(kmin=-12,kmax=20,ebin=1.0d0)  
parameter(nswEEP=100000,nup=5000)  
parameter(temp=1000.0)
```

After compilation of SMMP with the make command and running the program using the script 'smmp.cmd' the calculated multicanonical parameters are written into the file 'muca.d'. For further iteration of these parameters additional quantities such as the accumulated histogram are written into a file 'mpar_full.d' (out of which 'mulcan_par' reads them if 'l_iter=.true.' is set). We show here only the file 'muca.d' which lists the index of the energy bin, the $\beta(E)$ and the $\alpha(E)$ parameters (see [1] for more detailed explanations and references on multicanonical simulations):

```
-12  7.99001526  41.8549769  
-11  7.99001526  41.8549769  
-10  7.99001526  41.8549769  
-9   4.16266162  7.40879417  
-8   3.1249691  -0.892745986  
-7   2.44799324 -5.63157705  
-6   2.25973519 -6.76112535  
-5   1.98376871 -8.14095772
```

```

-4  1.80583549 -8.85269062
-3  1.69241238 -9.19295995
-2  1.65228855 -9.27320761
-1  1.60274413 -9.32275203
0   1.53176311 -9.32275203
1   1.47795021 -9.26893914
2   1.42184662 -9.15673195
3   1.2908957  -8.76387918
4   1.24563637 -8.58284188
5   1.15611832 -8.13525162
6   1.15781701 -8.14544378
7   1.01678513 -7.15822063
8   0.953895628 -6.65510459
9   0.855892018 -5.7730721
10  0.847746885 -5.69162077
11  0.738656239 -4.49162366
12  0.654146664 -3.47750877
13  0.644605311 -3.35347117
14  0.553037646 -2.07152387
15  0.474305513 -0.890541874
16  0.490066347 -1.14271521
17  0.409995341  0.218491892
18  0.422133779 -1.08246745E-15
19  0.422133779 -5.55111512E-16
20  0.422133779  0.

```

A test simulation of 100,000 Monte Carlo sweeps (using subroutine '*mulcan.sim*' with these weights led to the following histogram that is flat over the studied energy range (as is characteristic for the multicanonical ensemble) (note that the last energy bin contains all entries for energies larger/equal than 20 Kcal/mol):

```

-12  0
-11  0
-10  2515.
-9   1005.
-8   1036.
-7   1014.
-6   1113.
-5   1200.
-4   1549.
-3   2193.
-2   2665.
-1   2923.
0    2901.
1    3098.

```

2 3120.
3 3025.
4 3083.
5 3183.
6 3216.
7 3122.
8 3170.
9 3261.
10 3338.
11 3340.
12 3307.
13 3515.
14 3434.
15 3585.
16 3613.
17 3538.
18 3462.
19 3315.
20 17147.

9.4 Regularization of the B domain of protein A

Frequently, one wants to compare the results of a simulation with experimental data. When comparing properties of calculated configurations with that of structures from the Protein Data Bank (PDB) one has to remember that the actual bond lengths and angles in the PDB-structure will slightly differ from the fixed values that are assumed with the ECEPP or FLEX potentials. Forcing the molecule into the standard bonding geometry model of ECEPP (or FLEX) may lead to un-physically high energies. The process of finding an optimal structure within the standard geometry model starting from a PDB-structure is called regularization.

Our example we concentrate on the regularization of the atom coordinates for residues 10-55 of the B domain of *Staphylococcus Aureus* Protein A, which were taken from entry *1BDD* of the Protein Data Bank (see <http://www.rcsb.org>), as provided with this package in file *1bdd.pdb* (in sub-directory '*EXAMPLES*'). Setting the variable '*iabin = 0*' in *main.f* makes *init_molecule* call subroutine *pdbread* that reads the amino acid sequence and the atomic coordinates from the PDB-structure and stores this data in arrays declared in '*INCP.H*'. Then *pdvvars* is automatically called to measure all dihedral angles. In order to regularize the PDB-structure one has to: call *regul(1)* as the task subroutine in '*main*' (replacing call *minim(1)* as provided with program package). Set the following parameters in '*main*', accordingly:

```
flex = .false.  
sh2 = .false.  
epsd = .false.  
itysol = 0  
grp_n = 'nh2'  
grp_c = 'cooh'  
iabin = 0
```

Note that '*regul*', as any other task subroutine, has to be called AFTER '*init_molecule*' and that '*nml*' has to be set to 1 in the present version of SMMP.

Re-compile and link the package with this modified *main.f* and use the following script (i.e. file *smmp.reg* in sub-directory '*EXAMPLES*')

```
cd ..  
./smmp <<!  
./EXAMPLES/1bdd.pdb  
!
```

First '*regul*' obtains its input from the arrays declared in '*INCP.H*'. In a first step a naive representation of the molecule is build in the ECEPP geometry from the stored dihedral angles that were calculated from the PDB coordinates. This simple model serves as start configuration for the regularization. In our example, its root-mean-square deviation (over all heavy atoms) compared to the PDB-structure initially is 2.8 Å and its ECEPP-energy $\approx 10^{10}$ kcal/mol, mainly due to excessive van-der-Waals repulsions. The regularization

starts by first minimizing a term that measures the sum of squared distances between heavy atoms in the SMMP-structure and the given PDB-structure. This leads to a rmsd of 0.14 Å kcal/mol. After this initial step a list of bad contacts (vdW-energy of more than 2 kcal/mol) in the SMMP-structure is printed out. In a second step the physical energy is minimized, allowing only the free hydrogens to move, and the bad contacts and rmsd are displayed again. This step reduces the ECEPP-energy to $\approx 10^3$ kcal/mol. '*Regul*' aims at further reducing this energy while at the same time keeping the rmsd as small as possible. This is done by minimizing a composite energy from the weighted sum of physical ECEPP-energy and constraint energy (the quadratic distance measure) over 10 iterations. In each iteration the weight of the constraint energy is successively lowered (from 1 to 0) and the weight of the ECEPP energy raised (from 0 to 1). At the end, the dihedral angles of the final structure are printed out together with its rmsd and a list of (eventually) remaining bad contacts.

The following output is obtained from this last iteration with a weight of 1 for the ECEPP-energy and a weight of 0 for the constraint energy, that leads to a SMMP-configuration with a total ECEPP-energy of -430.8 kcal/mol and a rmsd of 1.76 Å, compared to the PDB-structure:

Energy BEFORE minimization:

Total: -0.40128E+03

Coulomb: -0.4687E+02 Lennard-Jones: -0.3315E+03 HB: -0.6948E+02

Variables: 0.4654E+02 Solvation: 0.0000E+00 Regularization: 0.3407E+03

```

Step    1: energy  0.133729E+16  ( 0.174001E+37, 0.333203E+12 )
Step    2: energy  0.363055E+04  ( 0.368838E+12, 0.989962E+11 )
Step    3: energy -0.274981E+03  ( 0.648848E+08, 0.274633E+11 )
Step    4: energy -0.333018E+03  ( 0.379334E+09, 0.889472E+09 )
Step    5: energy -0.399762E+03  ( 0.582647E+06, 0.398837E+08 )
Step    6: energy -0.401426E+03  ( 0.194084E+05, 0.100221E+08 )
Step    7: energy -0.401658E+03  ( 0.309779E+05, 0.129744E+08 )
Step    8: energy -0.402059E+03  ( 0.181490E+05, 0.213119E+08 )
Step    9: energy -0.402563E+03  ( 0.172799E+06, 0.772566E+08 )
Step   10: energy -0.402764E+03  ( 0.330004E+06, 0.164070E+09 )
.....
.....
.....
Step  420: energy -0.430829E+03  ( 0.962300E-04, 0.866792E+10 )
Step  421: energy -0.430829E+03  ( 0.122085E-04, 0.866791E+10 )
Step  422: energy -0.430829E+03  ( 0.305390E-05, 0.866790E+10 )
Step  423: energy -0.430829E+03  ( 0.579932E-06, 0.866791E+10 )
Step  424: energy -0.430829E+03  ( 0.633518E-07, 0.866791E+10 )
Step  425: energy -0.430829E+03  ( 0.158450E-07, 0.866791E+10 )
Step  426: energy -0.430829E+03  ( 0.894306E-09, 0.866791E+10 )

```

Step 427: energy -0.430829E+03 (0.143664E-09, 0.866791E+10)
 Step 428: energy -0.430829E+03 (0.143664E-09, 0.866791E+10)
 ---- CONVERGENCE ----

Final energies -----

Total: -0.43083E+03

Coulomb: -0.5354E+02 Lennard-Jones: -0.3531E+03 HB: -0.7209E+02

Variables: 0.4795E+02 Solvation: 0.0000E+00 Regularization: 0.5382E+04

.....

RMSD = 1.75752506

Dihedral angles of the regularized structure;

1 : x1 : -84.1
 1 : x2 : 65.8
 1 : x3 : 76.8
 1 : x4 : 0.3
 1 : phi : -110.1

.....

46 : psi : -30.9
 46 : omg : 177.2
 46 : x1 : -177.1
 46 : phi : -78.5
 46 : pst : -37.6
 46 : omt : 178.5

9.5 Parallel Tempering simulation of Met-enkephalin

In the following example, a Parallel Tempering simulation of Met-enkephalin in gas-phase is performed that relies on the ECEPP/3 force-field and where all dihedral angles are free. References for the algorithm can be found in [1]. The simulation is performed on a parallel computer and 6 nodes are used. It is best to use a separate Makefile (named '*Makefile_p*' in our program distribution) in which one has to set the correct compiler and linker option for the MPI-installation of the specific parallel computer. Note that the main program is no longer '*main*' but '*pmain*' in which one now has to set the following parameters accordingly:

```
no = 6
flex=.false.
sh2=.false.
epsd=.false.
itysol=0
grpn = 'nh2'
grpc = 'cooh'
```

After reading the amino acid sequence and start configuration, and initialization (by means of calling '*init_energy*' and '*p_init_molecule*'), '*pmain*' calls '*partem_p*' which does the parallel tempering simulation. The subroutine reads as input the distribution of temperatures and for measuring purposes a map of contacts from external files (in our example '*enkefa.temp*' and '*enkefa.ref*' in sub-directory '*EXAMPLES*') that have to be provided.

In our example, the simulation starts from random configuration with 10,000 sweeps for equilibration. The molecule is then studied over 100,000 sweeps on each node, with measurements taken every 10 sweep. Hence the following parameters have to be set

```
parameter(nequi=10,nswp=1000,nmes=10)
parameter(newsta=.true.)
```

After compilation ('*Makefile_p*') the program is ready to run. We do not provide a script as its form depends on the specific MPI installation. The data are written into a file '*ts.d*'. In our case, the data are the sweep, the temperature index, the index of the replica with that this temperature is currently associated, the inverse temperature, and energy, radius-of gyration, number of residues that are part of a helix or sheet, number of hydrogen bond, total number of contacts and the number of contacts that appear also in the reference configuration:

10	1	3	0.50	12.97	5.68	0	2	0	0	0
10	2	2	0.84	16.00	6.46	1	0	0	0	0
10	3	6	1.26	4.14	4.92	3	1	0	1	1
10	4	1	1.68	4.79	4.74	0	1	0	1	1
10	5	5	2.01	1.31	4.48	0	0	0	1	1
10	6	4	2.52	-1.03	4.43	0	0	0	1	1

.....										
.....										
.....										
.....										
100000	1	2	0.50	20.39	5.47	0	0	0	0	0
100000	2	4	0.84	11.05	4.81	0	1	0	1	1
100000	3	3	1.26	10.28	5.16	0	0	0	0	0
100000	4	6	1.68	-3.06	4.62	2	0	0	1	1
100000	5	1	2.01	-4.01	4.75	1	0	0	1	1
100000	6	5	2.52	-7.71	4.69	1	0	0	1	1

At the end of the simulation, the final configurations are written into external files (in our case '*enkefa.oxx*' where 'xx' marks the replica index. Further information for re-starts is written into '*par-R.in*' that also collects statistics on the run.